APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DATA STRUCTURE FOR FAST CASE-SENSITIVE AND INSENSITIVE SEARCH

APPLICANT: HOLGER SCHWEDES

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 399 314 076 US

March 31, 2004

Date of Deposit

DATA STRUCTURE FOR FAST CASE-SENSITIVE AND INSENSITIVE SEARCH

BACKGROUND

[0001] When using a search engine or similar technology to search in an information system for a certain term, which may be either a single word or a phrase, a user enters a string of characters. The string of characters may include a specific combination of uppercase and lowercase letters.

[0002] Where letters are used for the search term, a number of cases are possible. In one case, the user wishes to retrieve only documents containing the search term with the specific combination of uppercase and lowercase letters. In this case, the user wishes to perform a "case-sensitive" search. In another case, the user wishes to retrieve all documents containing variants of the search term with any combination of uppercase and lowercase letters. In this case, the user wishes to perform a "case-insensitive" search. The search engine should enable the user to choose which kind of search to perform, without sacrificing speed or efficiency.

[0003] In one example, a search term includes a string of characters. In the search system, a dictionary stores a list of all acceptable terms for a string attribute. Logically,

the dictionary is a sorted list of strings. In the example, let the dictionary include the set of terms {"ADAM", "Adam", "adam"}. Corresponding search results are exemplified below:

A case-sensitive search for	"Adam"	finds	{"Adam"}.
A case-sensitive search for	"AdAM"	finds	{}.
A case-sensitive search for	"adam"	finds	{"adam"}
A case-insensitive search for	"Adam"	finds	{"ADAM", "Adam", "adam"}
A case-insensitive search for	"AdAM"	finds	{"ADAM", "Adam", "adam"}
A case-insensitive search for	"adam"	finds	{"ADAM", "Adam", "adam"}

[0004] Conventional techniques for implementing case-sensitive searches are fast and efficient. However, techniques for implementing case-insensitive search on the same dictionary may be much slower and less efficient, since standard dictionary orderings of terms in the index for a document collection may not group variants of terms with different uppercase and lowercase spellings together. Case-insensitive search on a different dictionary can be fast, but this requires maintenance of two separate dictionaries, which is inefficient.

[0005] There exists a need to raise the speed and efficiency of case-insensitive search by defining a dictionary implementation that enables comparably fast sensitive and insensitive search to be performed on the same list of terms in a dictionary.

SUMMARY

[0006] This document discloses a method and system to define a dictionary sorting function that orders terms case-insensitively into blocks that are equivalent except for case variations, then order blocks of equivalent terms such that variants with uppercase letters precede variants with lowercase letters.

[0007] In accordance with an embodiment, a method of fast case-sensitive search of a dictionary using one or more search terms is disclosed. The dictionary includes an ordered list of terms. The method includes setting a dictionary sorting function to sort the ordered list of terms based on case sensitivity, and determining, according to the dictionary sorting function, whether a term corresponding to a search term is in an upper or lower half of the ordered list. The method further includes selecting an upper or lower half of the ordered list that includes the search term.

[0008] In accordance with another embodiment, a system for fast case-sensitive search of the dictionary includes a search engine configured to receive a user search query for a search of the dictionary, and to return a search result list. The search engine is further configured to enable a user to select whether to perform a case-sensitive or case-insensitive search

of the dictionary. The system further includes an ordering module configured to order the terms in the dictionary based in part on the binary numbers corresponding to the ASCII coding of alphanumeric characters comprising the terms in the dictionary.

[0009] The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] These and other aspects will now be described in detail with reference to the following drawings.

[0011] FIG. 1 shows an example of a system for fast case-sensitive or case-insensitive search of a dictionary.

[0012] FIG. 2 illustrates one example method of a case-sensitive search.

[0013] FIG. 3 illustrates one example method of a case-insensitive search.

[0014] FIG. 4 illustrates another example method of a case-insensitive search.

[0015] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0016] A method and system is disclosed that defines a dictionary sorting function that orders terms case-insensitively into blocks that are equivalent except for case variations. Then, blocks of equivalent terms are ordered such that variants with uppercase letters precede variants with lowercase letters. In one implementation, the dictionary includes appropriate case-oriented spelling variants of each term.

[0017] FIG. 1 shows a system 100 that includes a search engine 102 configured to receive a user search query for a search of a dictionary 106 and to return a search result list. In accordance with an exemplary embodiment, the search engine 102 is further configured to enable a user to select whether to perform a case-sensitive or case-insensitive search of the dictionary 106. The dictionary 106 is a stored list of all terms for a string attribute. Logically, the dictionary 106 is a sorted list of strings. An ordering module 104 is configured to order the entries in the dictionary 106 similar to a numerical sort on the basis of the binary numbers corresponding to the ASCII coding of alphanumeric characters,

and in particular the binary differences between upper and lower cases of the ASCII codes of those alphanumeric characters.

[0018] A resulting order of the ordering module 104 includes a dictionary sorting function LT that is case-insensitive, with blocks of equivalent case-insensitive words. For blocks of equivalent case-insensitive words, the order can be uppercase letters before lowercase letters. Table 1 illustrates one type of LT ordering of an example result for the word "ADAM." In practice, the dictionary 106 can include only those case variants that actually appear in the indexed documents, on not every possible variant of a word.

...	ADAM
adA	aDAm
ada	aDaM
...	aDam
adal	adAM
ADAM	adAm
ADAm	adaM
AdaM	adam
Adam	ADAN
AdAM	...
AdAm	ADAMA
AdaM	ADAMa
Adam	...

TABLE 1

[0019] The dictionary sorting function LT is defined for two strings x and y such that: LT(x, y) if and only if x precedes y in a dictionary ordering. Various implementations of the dictionary sorting function LT are possible using various programming languages or techniques. In "infix"

notation: x < y iff x precedes y in the dictionary, and LT(x, y) iff x < y.

[0020] Typical ASCII sorting does not list the different spellings of a term together in the dictionary, so it cannot be used to perform a fast search. Example:
"ADAM" < "BOBBY" < "adam"
Accordingly, the dictionary sorting function LT includes a parameter sensitive with possible values true and false. If sensitive=false, all case variants of "Adam" are equivalent under LT.

[0021] In an exemplary embodiment, a case-sensitive search is performed as a normal binary search in a list of all terms. FIG. 2 shows a case-sensitive search. At 202, a determination is made whether a search term should be in upper or lower half of the list. At 204, the upper or lower half is selected. At 206, a determination is made whether the search term is in the upper or lower half of the selected half. At 208, the upper or lower half of the half is selected. These steps are repeated until there is only one term in the selected half. If there is only one term in the selected half, then at 210, a determination is made whether the one term is the search term. If yes the method ends at 212. To compare terms, the function LT is used with parameter sensitive=true.

[0022] In an alternative exemplary embodiment, a case-insensitive search can be performed in several ways. FIG. 3 shows a first method 300 for a case-insensitive search. At 302 a binary search for the start of the insensitively equal terms in the dictionary listing is executed. To compare terms, the function LT is used with parameter sensitive=false. At 304, for each term an evaluation is made whether it is insensitively equal to the search term. If so, at 306, that term is added to the result list. If not, a next term is evaluated at 308. While each term can be evaluated one-by-one, these steps may be performed on blocks of two or more terms.

[0023] FIG. 4 shows a second method 400 for a case-insensitive search. At 402 a binary search for the start of the insensitively equal terms in the dictionary listing is executed. To compare terms, the function LT is used with parameter sensitive=false, as with 302 in FIG. 3. At 404, a binary search in the dictionary is made for each term to determine the last term X that is still insensitively equal to the search term. Given a dictionary ordering of uppercase before lowercase, at 406 the last term X is obtained from the search term by converting the search term into lowercase letters. All terms between the start found in 402 and the term X found at 406 are the result set. As with method 300,

these steps may be performed on blocks of two or more terms in parallel.

[0024] In sum, the function LT ensures that case-insensitively equal terms stand together in the dictionary: terms are first compared insensitively. If case-insensitively equal, they are then compared case-sensitively. Where n = number of terms in the dictionary (which may be many millions), k = average number of characters in a search string (a small fixed number, such as 10), and m = number of different spelling variants for a term (where maximum value is about 2^k, such as 1000), the case-sensitive search described at 200 above yields a result: O(log2(n) * k) ~~> O(log(n)).

[0025] The case-insensitive search of method 300 yields a result: O(log2(n) * k + m * k) ~~> O(log(n)). The case-insensitive search described with respect to method 400 yields a result: O(2 * log2(n) * k) ~~> O(log(n)).

[0026] Although a few embodiments have been described in detail above, other modifications are possible. Rearrangement of the logic flows depicted in FIGS. 2-4 are within the scope of the embodiments described therein. Other embodiments may be within the scope of the following claims.